(2)

RL-TR-92-122
Final Technical Report
June 1992

DESIGN FOR RELIABILITY - CURRENT DENSITY

University of Illinois at Urbana-Champaign

Ibrahim N. Hajj, Vasant B. Rao, Richard Burch, Hungse Cha, Russell Iimura, Harish Kriplani, George Stamoulis

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Systems Command
Griffiss Air Force Base, NY 13441-5700

RL-TR-92-122 has been reviewed and is approved for publication.

APPROVED: Martin J. Walter

MARTIN J. WALTER
Project Engineer

FOR THE COMMANDER: John J. Bart

JOHN J. BART
Chief Scientist, Reliability Sciences
Electromagnetics & Reliability Directorate

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE June 1992	3. REPORT TYPE AND DATES COVERED Apr 90 - Jul 91

4. TITLE AND SUBTITLE DESIGN FOR RELIABILITY - CURRENT DENSITY	5. FUNDING NUMBERS C - F30602-88-D-0028, Task N-0-5700 PE - 62702F PR - 2338 TA - 01 WU - P1
6. AUTHOR(S) Ibrahim N. Hajj, Vasant B. Rao, Richard Burch, Hungse Cha, Russell Iimura, Harish Kriplani, George Stamoulis	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Illinois at Urbana-Champaign Coordinated Science Laboratory 1101 W. Springfield Ave Urbana IL 61801	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory (ERDD) Griffiss AFB NY 13441-5700	10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-92-122

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Martin J. Walter/ERDD/(315) 330-4102

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This work describes the computation of expected and variance current density waveforms in sections of the power/ground bus for the purpose of estimating MTF due to electromigration. This work involves three main tasks: 1) Further development and enhancement of probabilistic simulation methods for computing current and voltage waveform statistics; 2) development of hierarchical extraction software for the generation of the circuit netlist and parasitics from hierarchically described layouts; 3) computation of expected value and variance current density waveforms in the RC model of the bus. The probabilistic approach for computing current and voltage statistics has been made more general and refined especially as related to the variance of the current waveforms. The extraction software reads hierarchically described layouts, and generated circuit descriptions with the same hierarchy. An approach for computing the expected value and variance of the current density for electromigration evaluation has been implemented which uses network reduction and sparse matrix techniques. The advances are described and applied to test cases for evaluation.

14. SUBJECT TERMS Reliability, Probabilistic simulation, electromigration, current density, extraction, VLSI CMOS circuits	15. NUMBER OF PAGES 52
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

FINAL REPORT EVALUATION

VLSI Design For Reliability - Current Density

The growth in the use of custom, semi-custom, and application specific ICs (ASICs) in AF systems has brought several changes to the reliability qualification process. This occurred because the costs associated with reliability characterization has become a major part of the device cost when there are only a few hundred or thousand of such devices produced. Further, the cost of a second or third pass at the design/fabricate/test/evaluate reliability cycle is very time consuming as well as expensive.

Analyzing an IC design for susceptibility to life limiting failure mechanisms is one approach to enhance the probability that at the end of the first pass of the design/fabricate/test/evaluate reliability cycle, the device will have an acceptable reliability.

This effort describes a more accurate and general approach to calculating the expected value and variance of the currents in a digital CMOS circuit, enhancements to extraction of the metal RC networks and circuit descriptions appropriate for probabilistic simulation, and the computation of current density waveforms in the bus metal. These calculations address susceptibility to electromigration, and may be extendible to power estimation and IR drop problems.

Other efforts sponsored by Rome Lab in the area of design for reliability include simulation methods for hot electron degradation in CMOS ICs, where realistic charge distributions are employed in modeling the localized oxide damage.

[illegible]

MARTIN J. WALTER
Design & Diagnostics Branch
Microelectronics Reliability Division

DESIGN FOR RELIABILITY-CURRENT DENSITY

Final Report

Contract F30602-88-D-0028(Task N-0-5700)

Abstract

This report describes the work accomplished during the period April 1, 1990, to July 31, 1991, on reliability analysis of VLSI CMOS circuits for electromigration prevention. The work is a continuation of research on the same subject that was supported under the same contract during the previous year. The main emphasis of the work is the computation of the expected value and variance current density waveforms in sections of the bus for estimating mean-time-to-failure (MTF) due to electromigration effects, as well as to determine the width of the bus sections in order to keep the MTF above a certain safe limit. The work involved three subtasks: (1) Further development and enhancement of the probabilistic simulation method to compute the current and voltage waveform statistics; (2) development of hierarchical extraction software for extracting transistor circuit description and parasitics from hierarchically specified CIF layout specifications; and (3) computation of the expected value and variance current density waveforms in the RC model of the bus, given the expected value and variance current waveforms at the bus contacts.

In probabilistic simulation, the statistical approach for computing the current and voltage statistics has been refined and made more accurate and more general, especially in the derivation of the variance of the current waveform and the expected value of the gate delays. The new derivations are being implemented in a new hierarchical probabilistic simulator.

In the area of layout extraction, a hierarchical extractor has been developed and linked to the OCT/VEM design data base system. The extractor accepts transistor artwork circuit specified in CIF in hierarchical format and produces a circuit description with the same hierarchical format and produces a circuit description with the same hierarchy.

In the subtask on power and ground bus analysis, a numerical approach has been implemented to compute both the expected value and the variance current density waveforms in the bus for MTF estimation of electromigration effects. Previous methods have concentrated only on the expected value or considered only dc components of the current waveforms. The approach uses network reduction and sparse matrix techniques for fast solution.

DESIGN FOR RELIABILITY-CURRENT DENSITY

Final Report

Contract F30602-88-D-0028(Task N-0-5700)

1. Introduction

This task is concerned with the development of computer-aided design techniques, as well as, software tools for reliability analysis and design of Application Specific Very Large Scale Integrated (ASIC VLSI) CMOS circuits. There are many failure mechanisms that have to be considered when analyzing the reliability of integrated circuits. These include metal and contact electromigration, power supply voltage drop and noise, transistor degradation induced by hot-electron injection into the oxide, oxide breakdown, electrostatic discharge through input protection circuit, system-noise induced latchup, and charged-particle-induced soft errors.

Since wearout mechanisms leading to poor reliability are a result the circuit behavior over typically long periods of time, they depend *on the combined effects* of a large number of possible input waveforms rather than worst-case conditions. One of the main issues addressed in this task is the development of efficient techniques for representing the combined effects of all possible input waveforms by one or possibly two waveforms (expected and variance waveforms), and the derivation and implementation of fast techniques for generating these waveforms. This has led to the development of a new methodology which we call probabilistic simulation [1-2].

In this task, we concentrate our efforts on a specific reliability problem, namely, electromigration (EM). In [2], we showed that MTF due to electromigration depends on the expected value, or average, as well as the variance current density. Probabilistic simulation provides a very efficient way for obtaining these current density statistics without the need for having to carry out exhaustive simulation.

In our first annual report, we described in some detail our initial work on the probabilistic simulation approach and its application to compute current density and estimate MTF due to electromigration. In this second annual report, we describe our accomplishments during this past year. In section 2, we summarize our continued work on the probabilistic simulation approach. In section 3, we describe the work on hierarchical extraction; namely, the extraction of hierarchical transistor circuit specifications, including parasitics, from hierarchically specified layout CIF files. In section 4, we report on the development of sparse matrix techniques for the computation of density waveforms in the bus given the expected value and variance current waveforms at the contacts to the bus. In section 5, we summarize our results and discuss future work.

2. Probabilistic Simulation Techniques

Simply stated, probabilistic simulation attempts to compute current and voltage statistics, in the form of average or expected value and variance waveforms in a circuit for all possible input excitations, without the need of having to perform exhaustive simulation. Last year, we reported on the derivation of this new probabilistic simulation technique and its implementation in program CREST [1]. During this year, we have embarked on the development of a new hierarchical simulator in order to be able to analyze very large designs. In contrast, CREST accepts only flattened circuit descriptions. The new simulator keeps a hierarchical data structure that will allow hierarchical probabilistic simulation techniques to be implemented. In addition, we have proved and implemented new derivations for the probabilistic simulation technique. The new derivations, which are explained below, extend the previous ones and makes them more accurate. We first give a brief review of probabilistic simulation. The details of the new derivation and its implementation, together with comparisons with CREST results and with exhaustive SPICE simulation for a number of typical examples are included in a M.S. thesis report [3].

2.1 Probabilistic Waveforms

Probabilistic simulation can be described as an event-driven simulation related to switch-level simulation. As described in CREST [1] one must stop short of describing the input signals as stochastic processes. Therefore the probabilistic waveform description is chosen as the description of choice, which is somewhere between a deterministic waveform and its stochastic counterpart. Each input is characterized by the probability that it is *high*, which means that the corresponding node has high logic value. Also the times where switching occurs are given as deterministic variables. This description is not complete though. To establish a complete description of the input at every time point, the transition probabilities must be included in the description. Actually only the probability of one transition is required (we will use the probability that the input switches from low to high P_{lh} in the rest of this thesis), because given $P_h(t^-)$, $P_h(t^+)$, and $P_{lh}(t)$, then $P_{hl}(t)$ can easily be calculated from the relation

$$P_h(t^+) - P_h(t^-) = P_{lh}(t) - P_{hl}(t) \tag{2.1}$$

In this way using probabilistic waveforms we can describe a large number of inputs. An example of a probabilistic waveform is shown in the following figure. Using this example the use of a probabilistic waveform will be explained.

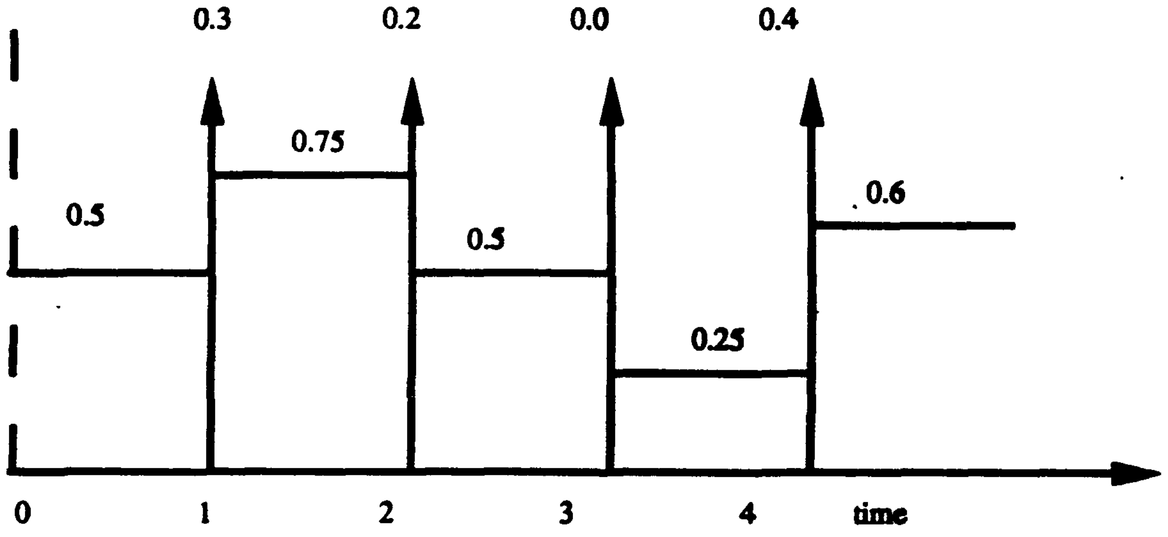

Figure 2.1. Example of probabilistic waveform

In Figure 2.1 the numbers over the arrows give the probability that a signal will switch from low to high, while the numbers on the flat part give the probability that the signal is high.

Now let us consider the time point t = 1. The probability that the input is high before t = 1 is 0.5, while the probability of being high after t = 1 is 0.75. The transition probability $P_{lh}(t)$ at that time point is 0.3. Applying the above formula in (2.1) the probability for a transition from high to low is $P_{hl}(t) = P_{lh}(t) - P_h(t^+) + P_h(t^-) = 0.05$.

2.2 Current Waveform Description

As has been stated above the objective is to derive the characteristics of the stochastic process describing the current through the bus. One complete description of the process would be to evaluate the current induced by every input vector, along with the probability of this taking place. Although this is possible (because we are dealing with a finite space) it can prove to be very time consuming. Consequently we have to limit ourselves to more incomplete descriptions. One such description involves the expected value waveform and the *variance waveform* (or R(t,t) of the corresponding stochastic process).

The expected value waveform represents the average value of all the paths at each time point. One can visualize it as the weighted sum of all the currents drawn by the circuit.

The variance waveform is calculated as $V(i(t)) = E[(i(t) - E[(i(t))])^2]$ at every time point. As it is becoming evident we will approximate the distribution of the current at each time point with a *gaussian* distribution. This is a simplification; but the accuracy of the results has validated our assumption.

The relationship between the expected value and the variance waveforms and MTF has been established in [2]. The remainder of this section will be on how to derive accurately and efficiently these two very important quantities given the input waveforms and the circuit topology.

2.3 Simulation Algorithm

The input waveforms given in the probabilistic waveform format are considered given and so is the description of the circuit. The objective is to calculate the waveform at the output node and the expected value and variance waveforms of the current through the gate. As it is known that transitions cause output events the simulation proceeds in an event driven fashion.

The circuit is first partitioned into an interconnection of channel-connected subcircuits. Simply stated, given a probabilistic representation of the voltage waveforms at the gates of the transistors in a subcircuit, the transistor conductances in the subcircuit become random variables. Graph reduction techniques are then applied to derive an equivalent probabilistic macromodel of the subcircuit with lumped output capacitances. The macromodel is then used to compute the output voltage probability waveform, the expected value and variance current waveforms drawn by the subcircuit, as well as the delay through the subcircuit. Probabilistic simulation has been implemented in CREST [1] and its validity has been established by simulating large industrial examples.

A key to the accuracy of the results of the probabilistic analysis of a subcircuit is the accuracy of the graph reduction step in the process of obtaining an equivalent macromodel. We derive an alternative reduction technique that is shown to be superior to the previous one, especially in the computation of the variance. The accuracy of this new method has been established by comparing the results to those obtained through exhaustive analysis.

2.4 Single Gate Reduction Algorithm

The analysis of a single channel connected partition of a circuit proceeds as follows. For a fully complementary gate we separate the p-part and the n-part and simulate them separately. Then we map the part we are simulating onto a graph as shown in Fig. 2.2, thus producing a simple non-directed graph.

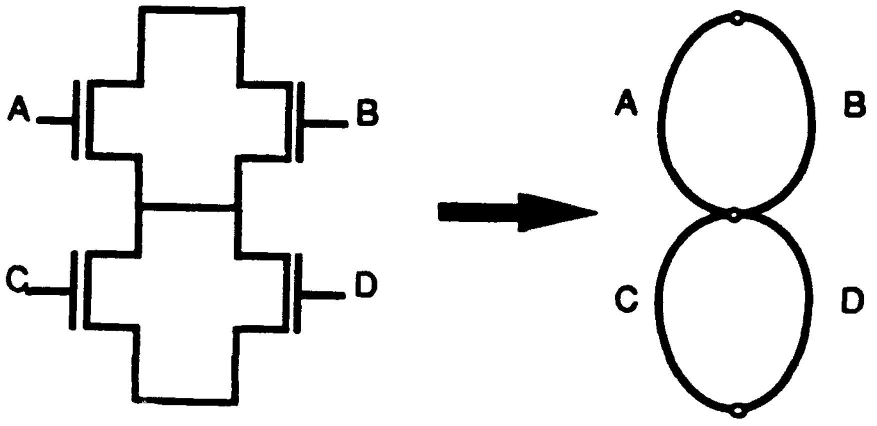

Figure 2.2 Mapping onto a graph

The objective of the whole process is to reduce the graph on the right to a single edge as shown in Fig. 2.3.

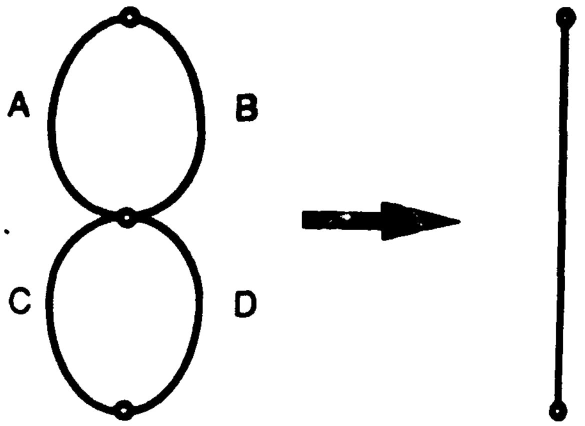

Figure 2.3 Reduction to an equivalent edge

In this part the procedure through which we derive the equivalent conductance of the circuit and the probabilities at all the necessary time points will be explained in detail.

We will consider only fully complementary CMOS circuits; thus the subcircuit can be further partitioned into a p-part and an n-part. Since these two parts are complementary, we will consider the analysis of the n-part only. Each transistor in the partition has a conductance associated with it when it is ON with value g_{on}. When the transistor is OFF it will have zero conductance. The probability of an nMOS transistor being ON is the probability of its gate node being "high" while the probability of a pMOS transistor being ON is the probability of its gate being "low." Thus the conductance of a transistor can be viewed as a random variable. The expected value and the variance of the conductance are defined respectively at any time t other than switching time as follows:

$$E[g] = g_{on} \times P_h(t) \tag{2.2}$$

$$V(g) = g_{on}^2 \times P_h(t) - (g_{on} \times P_h(t))^2 \tag{2.3}$$

As stated before we aim at finding an equivalent edge between the output node and the ground node (Vdd node for the p-part), so that we can calculate the current drawn at the ground contact. We assume for the time being that the circuit being eliminated can be eliminated in a series-parallel fashion, which is the most prevalent case. The probability that the voltage at the output node is high must also be calculated. We will consider the reduction of two basic cases: one is when two transistors are in parallel and the other when two transistors are in series. Assuming independent inputs, the parallel case becomes straightforward.

We will use the following notation in the sequel:

x,y:	the conductances of the two edges (transistors) to be reduced to one
g:	the equivalent conductance
P_{hx}:	probability that x is conducting
P_{lhx}:	probability that x is switching OFF to ON

It is also evident that when the equivalent edge is ON the output node is "low," since we are considering the n-transistor block.

In the parallel reduction case (shown in Fig. 2.4) the equivalent conductance is given by:

$$g = x + y \tag{2.4}$$

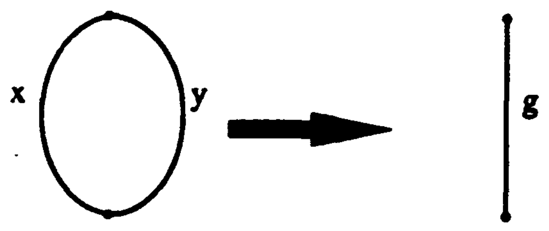

Figure 2.4 Parallel reduction

The stochastic quantities describing the equivalent edge are given by the following expressions, where the subscript p (for parallel) denotes those quantities associated with the equivalent edge. These results follow from simple probability theory and their proof is omitted.

$$P_{hp} = P_{hx} + P_{hy} - P_{hx}P_{hy} \tag{2.5}$$

$$P_{lhp} = P_{lhx}P_{ly} + P_{lhy}P_{lx} - P_{lhx}P_{lhy} \tag{2.6}$$

$$E[g] = E[x] + E[y] \tag{2.7}$$

$$V(g) = V(x) + V(y) \tag{2.8}$$

These equations are *not* approximations but exact expressions of the quantities for the equivalent edge.

The series case (shown in Fig. 2.5) is not as straightforward.

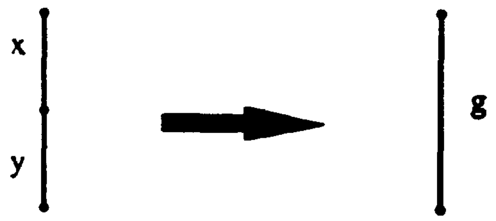

Figure 2.5 Series reduction

The probability that the equivalent edge is high and the probability that it switches are propagated by the following expressions which again follow from elementary probability theory (the quantities associated with the equivalent edge are here denoted by the subscript s).

$$P_{hs} = P_{hx}P_{hy} \tag{2.9}$$

$$P_{lhs} = P_{lhx}P_{hy} + P_{lhy}P_{hx} - P_{lhx}P_{lhy} \tag{2.10}$$

The above equations are also exact. But when it comes to propagating the expected value and the variance of the equivalent edge we have to resert to some approximation. In [1] the following expressions have been proposed:

$$\frac{1}{E[g]} = \frac{1}{E[x]P_{hy}} + \frac{1}{E[y]P_{hx}} \tag{2.11}$$

$$\frac{V(g)}{E[g]^4} = \frac{V(x)}{E[x]^4} + \frac{V(y)}{E[y]^4} \tag{2.12}$$

These expressions give only a rough approximation of the values in question. They can be greatly improved. As proposed in [8] a Taylor expansion of the expression (2.13) for the conductance of the equivalent edge for the deterministic case is obtained and by the appropriate algebraic manipulation a better approximation can be obtained. So starting from the following expression

$$g = \frac{x\,y}{x + y} \tag{2.13}$$

and inserting it to the following relations from [8] (p. 154)

$$E[g] = g + \frac{1}{2}\left(\frac{\partial^2 g}{\partial x^2}\sigma_x^2 + 2\frac{\partial^2 g}{\partial x\,\partial y} r\sigma_x\sigma_y + \frac{\partial^2 g}{\partial y^2}\sigma_y^2\right) \tag{2.14}$$

$$\sigma_g^2 = \left(\frac{\partial g}{\partial x}\right)^2 \sigma_x^2 + 2\left(\frac{\partial g}{\partial x}\right)\left(\frac{\partial g}{\partial y}\right) r\sigma_x\sigma_y + \left(\frac{\partial g}{\partial y}\right)^2 \sigma_y^2 \tag{2.15}$$

where σ_x^2 is the variance V(x), r is the correlation coefficient between x and y (zero in our case because we assume independence) and the right hand side is evaluated at (E[x],E[y]), we get

$$E[g] = \frac{E[x]E[y]}{E[x] + E[y]} - \frac{E[y]^2 v(x)}{(E[x] + E[y])^3} - \frac{E[x]^2 V(y)}{(E[x] + E[y])^3} \tag{2.16}$$

$$V(g) = \left[\frac{E[y]}{E[x] + E[y]}\right]^4 V(x) + \left[\frac{E[x]}{E[x] + E[y]}\right]^4 V(y) \tag{2.17}$$

Unfortunately these expressions do not give accurate results especially when P_h is rather small. But they are very important because the calculation of the expected value and the variance is decoupled from the correct propagation of the probability at each node. This can be helpful in cases where correlation hampers the propagation of the probabilities. But the fact that it cannot be used reliably in cases where P_h is very small limits the applications of these expressions. However by using conditional probabilities much better results are generated, but the correct propagation of the output event probabilities is a major factor affecting the accuracy of this method. To derive the new expressions, the expected value and the variance are calculated *given that the equivalent edge conducts.* Thus the following expressions are produced from (2.16) and (2.17):

$$E[g \mid x{\neq}0, y{\neq}0] = \frac{E[x \mid x{\neq}0]E[y \mid y{\neq}0]}{E[x \mid x{\neq}0] + E[y \mid y{\neq}0]} - \frac{E[y \mid y{\neq}0]^2 V(x \mid x{\neq}0)}{(E[x \mid x{\neq}0] + E[y \mid y{\neq}0])^3} - \frac{E[x \mid x{\neq}0]^2 V(y \mid y{\neq}0)}{(E[x \mid x{\neq}0] + E[y \mid y{\neq}0])^3} \tag{2.18}$$

$$V(g \mid x{\neq}0, y{\neq}0) = [\frac{E[y \mid y{\neq}0]}{E[x \mid x{\neq}0] + E[y \mid y{\neq}0]}]^4 V(x \mid x{\neq}0) + [\frac{E[x \mid x{\neq}0]}{E[x \mid x{\neq}0] + E[y \mid y{\neq}0]}]^4 V(y) \tag{2.19}$$

Where

$$E[x \mid x{\neq}0] = \frac{E[x]}{P_{hx}} \tag{2.20}$$

$$E[x^2 \mid x{\neq}0] = \frac{E[x]^2}{P_{hx}} \tag{2.21}$$

$$V(x \mid x{\neq}0) = E[x^2 \mid x{\neq}0] - E^2[x \mid x{\neq}0] \tag{2.22}$$

This method produces excellent results, but unfortunately it is dependent on the accurate propagation of the output probabilities. The reason that the second method works better than the first is the fact that the approximation for the first method is hampered by the presence of a large probability mass at zero. Clearly the Gaussian approximation is very poor. But the conditional distribution will follow exactly the actual distribution.

2.5 Calculation of Total Current and Delay

Having devised an algorithm for graph reduction is not enough for calculating the current drawn from the power bus by the circuit. We also need to model the current in such a way that the information that we get from the graph reduction is enough to fully describe the current waveform in the bus. Following the implementation in [1] the current in the bus will be described by two triangular pulses, one for the expected value and one for the variance. Therefore a peak value and an appropriate time span must be calculated in both these cases. Also the output event must have some delay associated with it for proper simulation of the circuit. In this chapter the method used in [1] will be compared with a new method that has been developed in this thesis.

The basic setup that is going to be used is shown in Fig. 2.6.

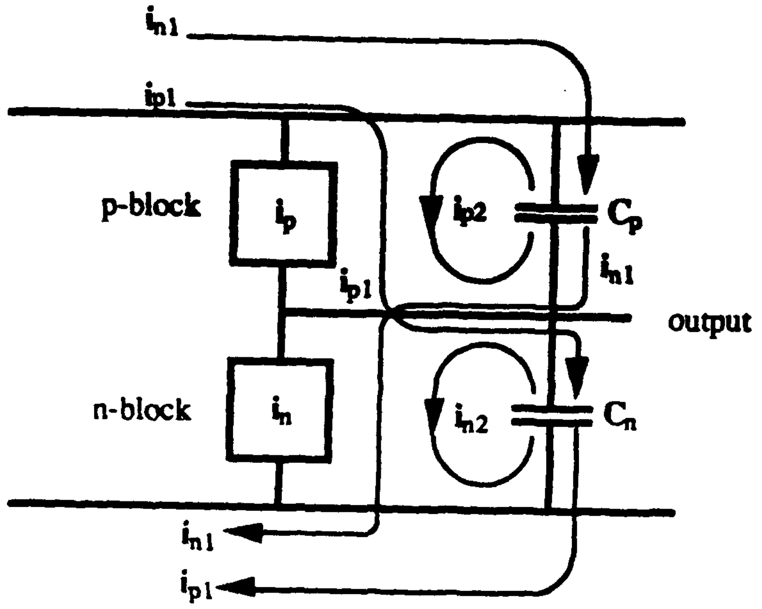

Figure 2.6 Model of a CMOS gate

CMOS circuits draw current from the bus only when their output changes state. Therefore the calculations for the current must be done when an input is switching. The method proposed in [1] requires the calculation of the probability that an edge is conducting given that the total conductance is zero for every edge in the graph that is being reduced. This requires a separate graph reduction for each edge, therefore the complexity of this algorithm is

$O(n^2)$. But as observed in [1] this is not very accurate because we cannot be certain that imposing the condition that the total conductance before switching occurs is zero will not affect the independence of the inputs. To ensure that the results are within some acceptable interval, bounds have to be imposed which compromise the accuracy of the method. For completeness we present the expressions for the current statistics as they are derived in [1]

$$E[i(t)] = E[i_p(t)] \times \frac{C_n}{C_p + C_n} + E[i_n(t)] \times \frac{C_p}{C_p + C_n} \tag{2.23}$$

$$E[i^2(t)] = E[i_p^2(t)] \times \left(\frac{C_n}{C_p + C_n}\right)^2 + E[i_n^2(t)] \times \left(\frac{C_p}{C_p + C_n}\right)^2 \tag{2.24}$$

with the quantities associated with $E[i_p(t)]$ and $E[i_n(t)]$ given by the expressions

$$E[i_p] = V_{dd} \times E[G_p(t^+) \mid G_p(t^-) = 0] \times P(G_p(t^-) = 0) \tag{2.25}$$

$$E[i_p^2] = V_{dd}^2 \times E[G_p^2(t^+) \mid G_p(t^-) = 0] \times P(G_p(t^-) = 0) \tag{2.26}$$

These are the results for the p-block. By replacing the p subscript with n we get the results for the n-part as well.

The new algorithm that is proposed in this thesis is simpler and more accurate than the one just described. It has been assumed that the inputs are *independent* and switch instantaneously. Since we make this assumption, then the probability that two inputs switch at the same time is zero, because if that was not the case there should be some correlation between the signals. Another very basic observation is that if the edge that switches is in parallel with an edge that is conducting, then switching will have no effect whatsoever to the output as illustrated in Fig. 2.7.

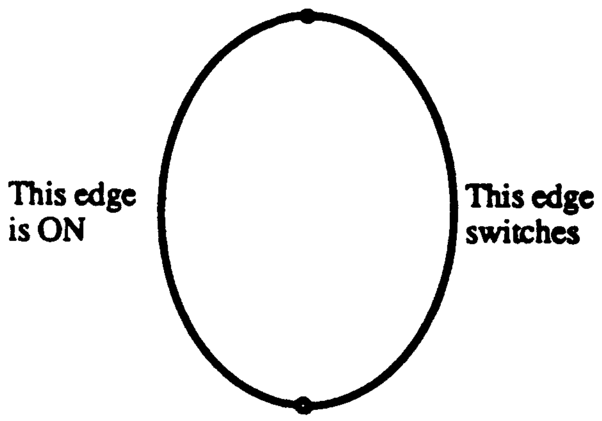

Figure 2.7 Illustration of the fundamentals of the new algorithm

It is evident that the edge that is ON effectively shorts the edge that switches, so that it has no effect on the rest of the circuit.

Given these observations we can state that for the output to switch, one of the edges must switch and *all* the conducting paths that are created *must* go through the edge that has switched. Now the calculation of the conductance at switching becomes very simple. Suppose that edge k is switching. Then for all other edges i

$$P(\text{i is ON} \mid \text{k is switching}) = P(\text{i is ON}) = P_{hi} \tag{2.27}$$

This result is due to independence. Since at switching all the current must pass through the transistor that is switching, then all paths that bypass that transistor should not conduct. To ensure this during elimination we use the following algorithm. In the description of the algorithm we split the edges into switching and non-switching. A switching edge is defined as the edge that switches and every equivalent edge that contains it. It is evident that because of independence there is only one switching edge in the graph that is being reduced. The only edge remaining after elimination is a switching edge. Every other edge is a non-switching edge.

1. Assign to the edge that switches probability 1.
2. Proceed to the elimination of the graph as described in Section 2.3.

   2a. Series reduction is performed as usual.

2b. In the parallel reduction we distinguish between two cases.

- If two non-switching edges are in parallel then elimination proceeds as usual.

- If one of the edges is switching then the non-switching edge in parallel with it is assigned $P_h = 0$ and the probability that it is not conducting is stored. Let the set of these edges be denoted by α.

3. Calculate E[I] and V(I) from the following expressions.

$$E[I_k] = V_{dd} \times E[G(\text{fromelimination})] \times P_{lhk} \times \prod_{i \varepsilon \alpha} P_{li} \tag{2.28}$$

and

$$E[I_k^2] = V_{dd}^2 \times E[G^2(\text{fromelimination})] \times P_{lhk} \times \prod_{i \varepsilon \alpha} P_{li} \tag{2.29}$$

The philosophy of this algorithm is that calculating the equivalent conductance in the way that was just described, one calculates the equivalent conductance given that the circuit will draw some current from the bus. Evidently there are more conducting paths than the ones calculated by this method, *but* the current drawn from those paths at the time of switching is zero (internal capacitances are considered negligible). So this method calculates the total current drawn by the circuit. The expressions just mentioned are derived very simply through the assumption of independent inputs, because the result of the elimination is the equivalent conductance given that an edge is switching and the edges that have been assigned probability zero are not conducting. A general example is presented in the following Fig. 2.8.

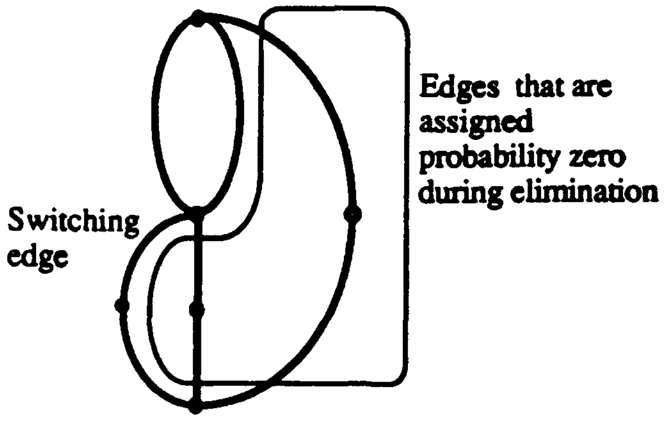

Figure 2.8 Elimination example

The algorithm proposed here is inherently much more accurate than the one proposed in [1] and is also O(n). This constitutes a great improvement over the previously used algorithm at a very critical step towards calculating the current.

Thus far the peak values for the expected value and variance waveforms have been calculated. For calculating the time span of both pulses the equations derived in [1] are sufficient enough for the proposed approximation. A brief description follows.

We define:

$$E[q] = V_{dd}C_nP_{lho} + V_{dd}C_pP_{hlo} \tag{2.30}$$

where P_{lho} and P_{hlo} are the probabilities that the output will go from "low" to "high" and vice versa. Also

$$E[iq] = \frac{V_{dd}C_n^2}{C_p + C_n} E[i_p] + \frac{V_{dd}C_p^2}{C_p + C_n} E[i_n] \tag{2.31}$$

The time spans for the expected value (τ_E) and the variance (τ_V) are given by the following simple relations.

$$\tau_E = 2 \times \frac{E[q]}{E[i]} \tag{2.32}$$

$$\tau_v = \frac{4}{3} \times \left(\frac{E[iq] - E[i]E[q]}{V(i)}\right) \tag{2.33}$$

For the delay of the output event a weighted average of the delay for the p-part and the delay for the n-part will be used. Starting from the deterministic delay model

$$t_d = \ln 2 \times \frac{C_p + C_n}{G} \tag{2.34}$$

and using simple probability theory we arrive at the following expressions for the delay of each of the transitions.

For the transition from low to high

$$E[t_{dp}] = \ln 2 \times (C_p + C_n) \times \left(\frac{1}{E[G_p]} + \frac{V(G_p)}{(E[G_p])^3}\right) \tag{2.35}$$

and for the transition from high to low

$$E[t_{dn}] = \ln 2 \times (C_p + C_n) \times \left(\frac{1}{E[G_n]} + \frac{V(G_n)}{(E[G_n])^3}\right. \tag{2.36}$$

where G_n and G_p are the equivalent conductances of the n-part and the p-part when they cause switching. For the total delay the following expression can be used.

$$E[t_d] = \frac{E[t_{dn}] \times P_{out,hl} + E[t_{dp}] \times P_{out,lh}}{P_{out,hl} + P_{out,lh}} \tag{2.37}$$

This expression combined with the approach that was used in its derivation, provides very accurate results, which are also easily manipulated (calculation of conditional probabilities and other related quantities is greatly simplified). That is so because the new elimination process breaks down the graph into independent edges connected in series.

In [1] some different expressions had been proposed but the straightforwardness and the simplicity of the previous expressions in conjunction with the results, introduce this method as a viable candidate for the calculation of the delay. The above expressions are simply the deterministic expressions where the conductance has been substituted by a random variable. According to [8] this is the best second order approximation. Further tests are needed to determine the best approach. We must note again that the probabilities used in the previous expression are the probabilities given that the output is switching through the part in question.

It should be made clear that this analysis is for one channel connected partition only and not for the entire circuit. Therefore the assumption of independence is valid to a great extent. As was demonstrated in [1] independence is a very safe approach. In the case of correlated inputs the techniques described in [1] can be used, but the stochastic part of the computation is based on independence and this is what has been presented here. As for the assumption that no two inputs of a channel-connected partition can switch at the same time, it is valid as long as the inputs are independent. But this can easily be relaxed. Let us assume that there are several signals switching at the same time. Because of independence each switching input can be handled separately and the total result can be derived by superimposing the partial results. It is very interesting to point out that this has already been done in [1] with excellent results. In any case this is not an essential part of this derivation because it was mainly used to avoid the cumbersome notation associated with two edges transitioning at exactly the same time, which was of no practical value anyway.

As for the cases where two current pulses are overlapping several heuristics can be derived in order to approximate the real current waveforms.

2.6 Results

In order to validate the methods proposed in the previous chapters, we appied the new formulas to many sample circuits and compared the results againsts the results of the previous methods and the results of a SPICElike simulation. In the first part of this chapter the new reduction scheme is compared against the previous one and in the latter part the delay calculation method is compared with the exact solution given by the SPICElike simulation. The gates tested include both simple and more complex structures.

First examples concerning the general elimination algorithm will be displayed. The results will be compared with the previously existing method and the SPICElike simulation. The eliminated graphs range from three transistor to multi-transistor layouts. Comparisons for two transistor structures (i.e., NAND and NOR gate configurations) have not been included because of their simplicity.

It must be noted that "method 1" is the method in which no signal probabilities are used in the derivation of the stochastic properties of the signals (other than evaluating the output probability and the characteristics of the transistors before elimination). The comparison is needed to calculate the effects of the degradation due to the normal approximation without conditional probabilities as explained in Section 2.3. The numbers associated with the arcs on the figures correspond to the probability that the edge conducts and its maximum conductance.

"Method 2" is the one in which conditional probabilities are taken into account and, as the examples will demonstrate, has been proven the most accurate among the three approximations.

Example 1.

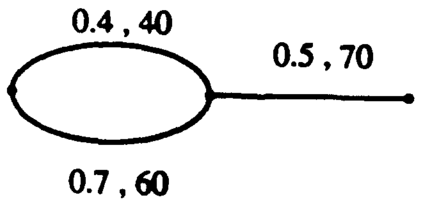

Figure 2.9 Example 1

	E[g]	V(g)
CREST	14.42	39.70
method 1	14.97	208.19
method 2	14.07	297.23
exact	14.08	297.29

Table 2.1 Results for example 1

This simple example highlights the basic notions that are going to become even more apparent in later examples. First of all method 2 is much more accurate than CREST in computing the expected value and the variance of the equivalent edge. This is something consistent throughout the comparison of the two techniques. Secondly method 1 is quite accurate but not as accurate as the other two. A third and final comment is that CREST underestimates the variance by a great amount which can lead in turn in to a great miscalculation of the current characteristics. As to the particular example, method 2 has shown great effectiveness in calculating this particular graph. For all the tested values for the probabilities as well as the conductances, the calculated values were within 1 percent of the values produced by the SPICElike "exact" simulation.

Example 2.

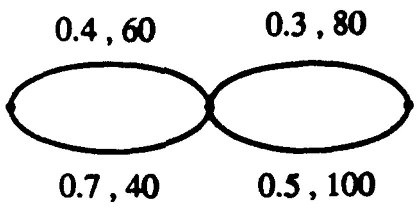

Figure 2.10 Example 2

	E[g]	V(g)
CREST	21.71	64.91
method 1	22.06	254.28
method 2	20.26	438.76
exact	20.41	433.89

Table 2.2 Example 2

This example consists of four transistors, which make the differences more apparent. The error for method 2 is much better than that of CREST both for the expected value and the variance. Also method 1 gives reasonable results for this case.

Example 3.

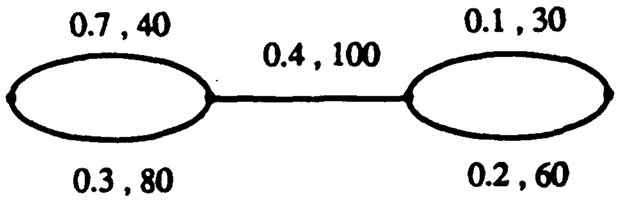

Figure 2.11 Example 3

	E[g]	V(g)
CREST	4.54	5.61
method 1	2.07	199.70
method 2	4.15	74.35
exact	4.17	74.83

Table 2.3 Example 3

In this example with five transistors the accuracy difference is more apparent. Method 1 fails because of the small probabilities involved in this example (i.e., the 0.1 and 0.2 conducting probabilities). Apart from that all the remarks made for the previous examples are valid for this one as well.

The final example consists of thirteen transistors and is characteristic of the behavior of the three formulae.

Example 4.

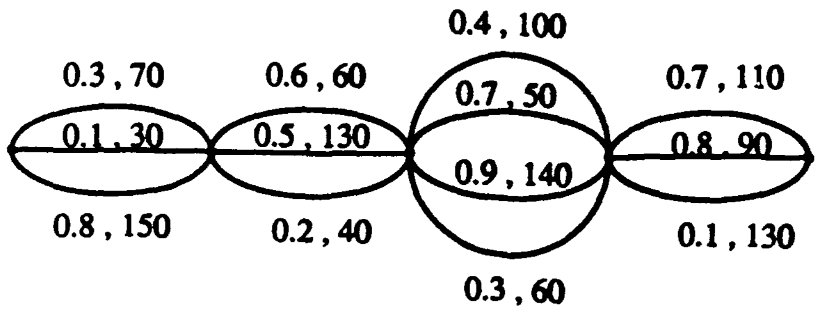

Figure 2.12 Example 4

	E[g]	V(g)
CREST	28.43	38.50
method 1	29.46	124.93
method 2	25.82	332.74
exact	25.15	349.50

Table 2.4 Example 4

The final conclusion from these examples is that method 2 is consistently better than CREST for calculating the expected value and the variance. The added precision in calculating the expected value is very important for the delay and current estimation, although not as spectacular as the enhancement in the calculation of the variance, which is also very important. The validity of method 1 has been demonstrated. Although it is not optimal it does not require the knowledge of the edge probabilities once the stochastic characteristics of each edge have been established. Finally a comprehensive table of results will be presented to better judge the new methods.

# of edges	m 2	CREST	m 1	exact
4	5.29	5.46	3.55	5.28
5	26.25	30.76	26.63	26.66
6	2.41	3.55	diverges	2.36
8	46.51	49.70	47.28	46.62
9	54.98	59.92	54.59	55.27
10	13.21	14.82	12.84	13.27
11	10.97	14.60	7.19	11.23
12	4.39	4.88	1.06	4.40
14	55.24	61.20	49.30	55.17
15	47.00	50.24	56.90	47.11
17	13.48	15.13	11.93	11.53
18	93.90	102.57	89.89	93.90
19	39.24	45.83	41.46	39.38
21	55.68	61.71	48.25	55.61
23	106.08	115.80	104.98	109.10

Table 2.5 Expected value examples

# of edges	m 2	CREST	m 1	exact
4	273.0	80.0	666.2	273.1
5	435.1	254.7	395.0	393.3
6	49.8	3.9	diverges	50.4
8	1282.8	606.4	1068.9	1272.3
9	1329.5	1232.6	1325.6	1335.9
10	243.1	79.8	268.4	244.5
11	248.3	118.3	461.3	255.5
12	78.6	7.5	150.3	79.4
14	1522.5	982.9	1758.0	1564.1
15	1308.3	631.4	890.9	1297.0
17	249.4	86.4	300.5	251.4
18	1914.6	1401.9	1838.4	1994.3
19	353.5	143.7	192.6	352.2
21	1543.7	1009.1	1859.1	1585.3
23	2154.3	1710.9	2052.3	2415.0

Table 2.6 Variance examples

Finally, we give a simple example of calculating the delay and the equivalent current. The example is a four transistor p-part in order to avoid the trivial cases of two or three transistors (for which the aforementioned quantities can be calculated exactly). The probabilistic quantities are specified so that only the p-part will conduct. So the p-part looks like the following figure:

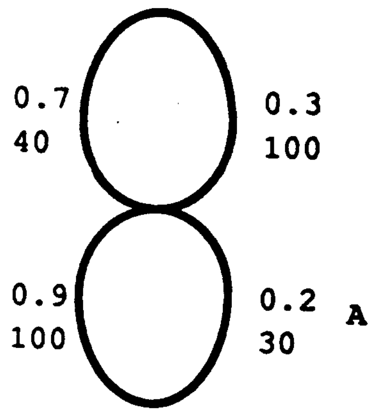

Figure 2.13 p-part of delay and current example

The probabilistic waveform going into input A is the following:

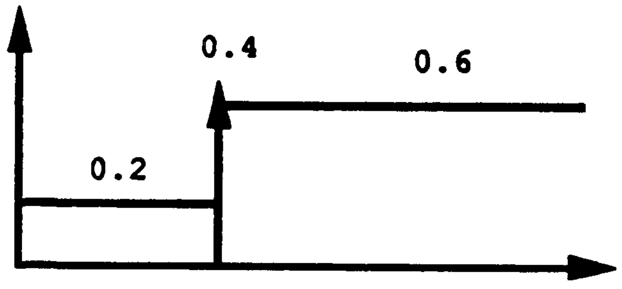

Figure 2.14 Input waveform for example in Fig. 2.13

From the analysis the following results have been produced by applying (2.28) to (2.37):

$$E[I_A] = 1.557,\ E[I_A^2] = 158.9,\ E[\text{delay}_A] = 36.42 \text{ time units}$$

The actual quantities as produced by exhaustive SPICElike analysis are

$$E[I_A] = 1.567,\ E[I_A^2] = 160.1,\ E[\text{delay}_A] = 35.96 \text{ time units}$$

3. Layout Extraction

Layout extraction is the transformation of artwork information into circuit information. It forms a very important link between the physical design and the design verification process as well as reliability analysis. In particular, computation of current density and electromigration estimation are closely linked to the physical process and the layout geometrical information. The input to an extractor consists of the mask geometries of the layout together with a process file. The artwork is typically described in CIF (Caltech Intermedicate Form), an ASCII format where each layout geometry is described in terms of its shape (usually a rectangle), mask layer, and coordinates. The output is typically in SPICE format which can be read by most circuit simulation programs.

In the previous year, we developed an extractor that was linked to the OCT/VEM design database system developed at the University of California at Berkeley. OCT is a database system capable of storing the design for an entire VLSI chip, including the schematic artwork, and the extracted netlist of a design. The basic unit of storage is a *cell*, which, for example may contain a single transistor, a logic gate, or an entire functional unit (ALU, CPU, or RAM). To efficiently represent the design information, cells are represented hierarchically, so that a cell may contain instances of other cells. In addition, parameters of the design may be annotated to the structural information, such as the dimensions of transistors, or parasitic capacitance values for nets.

VEM is the graphical editor for the OCT elements. Circuit schematics of artwork may be created and edited with VEM. The extractor that was developed during the first year of this work assumed flattened input format and produced flattened SPICE file. However, such a flattened representation is quite inefficient and is not practical for very large designs especially that designs are specified hierarchically. During this year, we have developed a new extractor, based on the previous one. The new extractor, called iCHARM, preserves and exploits the hierarchy of the input and produces a hierarchical circuit file as well as the power bus nets.

iCHARM is written in C, runs under UNIX* and has the following features:

CMOS circuits:

The extractor assumes the input layout is in CMOS technology with any number of metal interconnection layers. A technology file is read to determine the process parameters of each layer and the connectivity between layers.

Manhattan geometries:

iCHARM supports only rectangles as mask geometries; therefore, the extractor works only for *manhattan-style* layouts; i.e., layouts where the edges of the geometries are parallel to the x- or y- axis.

CIF and OCT input formats:

The input layout read by iCHARM is described in either the CIF (Caltech Intermediate Form) format or in the OCT database format.

Spice output format:

The extracted circuit description is output in the SPICE circuit simulator format. If the extraction is done hierarchically, each cell extracted generates a .SUBCKT card. An option to output the circuit in the OCT format also is envisioned.

Scanline algorithm:

The input layout is represented internally in iCHARM in terms of rectangles. A scanline algorithm is employed to carry out the extraction process. This is described in detail in this thesis.

The program may be run in several modes as follows:

*UNIX is a trademark of AT&T Corp.

(1) First, the user has the option of running the extractor in flat or hierarchical extraction mode. If there is a high degree of regularity in the circuit (if the circuit has just a few cells but perhaps a large number of instances), then the hierarchical mode should run faster and use less run-time memory than an equivalent run in the flat extraction mode.

(2) The user also has a choice of the parasitics of the interconnection lines that may be extracted. The default mode is to extract each net's substrate capacitance (the capacitance between the net and ground). In addition to the capacitance, the user may extract the resistance through each interconnection net.

(3) iCHARM has a special mode to extract the mask-level rectangles of the power bus nets (typically Vdd and GND). This mode does additional processing of the power bus nets for subsequent analysis by other programs to do reliability estimation.

A detailed description of the algorithms used in iCHARM together with the implementation issues are included in a M.S. thesis report [4].

In the following, we describe the results of extracting a large example, the 2uchp test chip, specified hierarchically in CIF.

The 2uchp design is a 10,000 transistor CMOS layout that was designed at the Rome Air Development Center of the U.S. Air Force. It is so named because the top-most cell is named "2uchp." The design was originally described in CIF, but it was converted to an equivalent Oct description to test the Oct input module of iCHARM.

The 2uchp chip allowed iCHARM to be tested with realistic layout situations. The chip consists of 74 different cells. The design hierarchy of the chip is shown in Figure 3.1. The major cells of the chip are a ram (*ram1*), rom (*rom2*), alu (*alun2*), miscellaneous logic (*PRNG*, *chp_qs*, *pr_leak*), and the padframe (*2ufrm*). Although most cells also contain mask geometries, in Figure 3.1 only the subcells are shown for each cell.

```
2uchp
    V, Cp
    chp_qs
    pr_leak
    2ufrm
    PRNG
    rom2
    alun2
    ram1

chp_qs
    Cp, Ca
pr_leak
    V, Ca
    pr_2_11
    leak_p
        Ca
2ufrm
    2u64p46
        sym2
            sym1
            sym3
                sym1
    X2IPD
        Cp, Ca
    X2TRI
        Cp, Ca
    PWR2
    OPAD2
        Cp, Ca
    GND2
```

```
PRNG
    V, Cp
    PRNGc
        Cp
        2350
            Cp, Ca
        1220
            Cp, Ca
        1830
            V, Cp, Ca
        1310
            Cp, Ca
    PRNGa
    V, Cp
        1310
            Cp, Ca
        tmp
            Cp
            1220
                Cp, Ca
            1830
                V, Cp, Ca
        2310
            Cp, Ca
        spacer
    PRNGb
    Cp
        tmp
            Cp
            1220
                Cp, Ca
            1830
                V, Cp, Ca
        2310
            Cp, Ca
        spacer
```

```
rom2
    V, Cp, Ca
    data
        Ca
        rom_ar2
            romcel2
                Ca
            wcnt
                Ca
    rom_rowda
        V, Cp, Ca
        rom_nor4
            Ca
        inv_add
            V, Ca
        rom_dec_el
            Ca
    deccol
        V, Cp
        1310
            Cp, Ca
        tri_8
            V, Cp, Ca
        1120
            Cp, Ca
        1130
            Cp, Ca
        1140
            Cp, Ca
    row_pul
        Cp, Ca
```

```
alun2
    Cp
    alur2
        1220
            Cp, Ca
        1310
            Cp, Ca
        1230
            Ca
        1240
            Cp, Ca
        1250
            Cp, Ca
        sp2
            Cp
        sp
    alur3
        1220
            Cp, Ca
        2310
            Cp, Ca
        sp2
            Cp
        sp
        1680
            Cp, Ca
        1660
            Cp, Ca
        1670B
            Cp, Ca
    2rlb
        Cp
        1220
            Cp, Ca
        1310
            Cp, Ca
        1230
            Ca
        1660
            Cp, Ca
        1670B
            Cp, Ca
    alun2s
        V
```

```
ram1
    V, Cp
    ram_rowd
        V, Ca
        ram_rowa
            V, Cp
            ram_dec_na
                Ca, Cp
            ram_nand
                Ca, Cp
        rowd_edg
    mem_blk
        ram_cel4
            ram_cell
                Ca, Cp
        blk_b_edg
        blk_l_edg
    col_blk_l
        V, Ca
        ram_cold
            ram_row
                Cp
                ram_dec_na
                    Cp, Ca
                ram_nand
            col_sens
                V, Cp, Ca
    col_blk_r
        V, Ca
        ram_cold
            ram_row
                Cp
                ram_dec_na
                    Cp, Ca
                ram_nand
            col_sens
                V, Cp, Ca
    ram_buff
        V, Cp
        1220
            Cp, Ca
        1310
            Cp, Ca
```

Figure 3.1 - The hierarchy of the 2uchp chip

Since the majority of the transistors are in the *ram1* cell, it is the most difficult cell to extract flat. In the *ram1* cell there are two instances of the *mem_blk* cell, which is a 512-bit static ram array.

The 2uchp chip contains a "hard" extraction case where cells overlap: the *2ufrm* cell is a padframe cell which overlaps the other major cells (*ram1*, *rom2*, *alun2*, *PRNG*, *chp_qs*, and *pr_leak*). Another major cell, the *alun2* cell, has overlapping instances. In *alun2*, the instance of *alun2s* overlaps the other instances over most of their area. Therefore to extract the entire chip hierarchically, it is important to avoid flattening the areas of overlap.

The cells of the chip were used to test the functionality of iCHARM. All of the cells were put through the flat extraction, parasitic extraction, power bus extraction, and hierarchical extraction modes of the extractor. Of most interest was the verification of the hierarchical extraction capability and the performance comparison of the hierarchical mode versus the flat mode. The test results for the larger cells of 2uchp are shown in Table 3.1.

Out of the 74 total cells, a number of leaf cells are simply connector or feed-through cells and do not contain any transistors, and a number of cells are simple logic gates (nand, nor, inv). To reduce the size of Table 3.1, none of these cells are reported. The second and third columns of Table 3.1 report the number of transistors and nets for each cell. Next, the ***regularity*** of a cell is defined as

$$\text{regularity} = \frac{\text{number of instances of the cell}}{\text{number of unique subcells}}$$

For instance, a cell may have one instance each of five subcells and four instances of another subcell which results in a regularity factor of 9/6 = 1.5. The number of overlap-rects of a cell is also reported in Table 3.1.

Table 3.1 - iCHARM Test Results with 2uchp Cells

Cell name	Tran-sistors	Nets	Reg-ularity	Overlap-rects	Flat Time	Flat Mem.	Hierarchical Time	Hierarchical Mem.
col_sens	16	18	50.0	26	0:00/0:00/0:03	46.7	0:00/0:01/0:04	84.6
ram_cell	6	8	7.0	0	0:00/0:00/0:00	9.5	0:00/0:00/0:00	12.8
ram_cel4	24	19	4.0	0	0:00/0:00/0:02	29.9	0:00/0:00/0:01	29.1
ram_nand	8	22	20.5	72	0:00/0:00/0:01	16.9	0:00/0:00/0:01	25.5
ram_dec_na	10	7	25.0	47	0:00/0:00/0:00	15.1	0:00/0:00/0:01	24.5
ram_rowa	28	39	2.0	57	0:00/0:00/0:02	46.4	0:00/0:01/0:03	69.6
mem_blk	3072	1122	58.7	1	0:00/0:07/4:47	3022.8	0:00/0:00/0:18	945.7
ram_rowd	448	145	18.5	18	0:01/0:02/0:39	502.0	0:01/0:01/0:11	393.3
alun2s	0	4	3.0	301	0:00/0:00/0:00	1.8	0:00/0:00/0:00	2.6
tri_8	14	16	40.0	1	0:00/0:00/0:01	28.3	0:00/0:00/0:02	57.0
rom_dec_el	4	8	9.0	10	0:00/0:00/0:00	4.4	0:00/0:00/0:00	6.0
inv_add	6	4	11.5	16	0:00/0:00/0:00	7.9	0:00/0:00/0:00	14.1
rom_nor4	24	26	70.0	51	0:00/0:00/0:02	27.1	0:00/0:00/0:01	39.6
romcel2	2	7	2.0	12	0:00/0:00/0:00	2.4	0:00/0:00/0:00	3.6
rom_ar2	256	305	80.0	370	0:00/0:00/0:13	178.9	0:00/0:00/0:03	207.8
row_pul	2	6	2.5	6	0:00/0:00/0:00	3.0	0:00/0:00/0:00	5.5
rom_rowda	110	55	17.7	49	0:01/0:01/0:07	106.3	0:01/0:01/0:05	134.9
data	256	49	128.5	75	0:01/0:01/0:19	226.6	0:01/0:03/0:15	459.5
PRNGc	47	28	4.6	4	0:01/0:01/0:06	76.7	0:01/0:01/0:06	72.2
OPAD2	23	5	172.5	4	0:01/0:01/0:03	96.4	0:01/0:02/0:07	149.7
X2TRI	34	10	215.0	1	0:01/0:01/0:04	124.7	0:01/0:03/0:09	179.4
X2IPD	16	5	126.0	1	0:00/0:01/0:02	74.5	0:00/0:01/0:04	105.3
chp_qs	2	8	16.0	5	0:00/0:00/0:00	8.9	0:00/0:00/0:00	15.1
2ufrm	922	176	9.0	1379	0:04/0:12/2:13	2429.7	0:04/0:09/3:32	6769.8
ram_row	18	27	1.3	58	0:00/0:00/0:02	33.0	0:00/0:00/0:02	54.3
ram_cold	50	53	1.5	69	0:01/0:01/0:10	108.1	0:01/0:01/0:07	140.1
ram_buff	12	9	3.5	54	0:00/0:00/0:02	35.4	0:00/0:00/0:01	35.9
col_blk_r	400	213	10.3	10	0:01/0:03/1:20	840.9	0:01/0:02/0:11	308.3
col_blk_l	400	213	10.3	9	0:01/0:03/1:20	840.8	0:01/0:02/0:11	307.9
2rlb	40	28	4.5	10	0:01/0:01/0:06	71.4	0:00/0:01/0:05	73.3
alur3	108	90	3.3	4	0:01/0:01/0:17	213.5	0:01/0:01/0:06	97.3
alur2	110	116	3.9	3	0:01/0:01/0:14	185.5	0:01/0:01/0:05	95.2
deccol	122	63	17.7	29	0:01/0:02/0:21	244.3	0:01/0:02/0:08	161.6
tmp	28	21	2.3	33	0:00/0:00/0:03	38.4	0:00/0:01/0:03	50.5
PRNGb	261	156	6.0	29	0:01/0:01/0:29	372.7	0:01/0:01/0:06	91.9
PRNGa	278	163	6.3	27	0:01/0:01/0:32	410.1	0:01/0:02/0:07	112.0
ram1	7404	2561	76.3	29	??	O.M.	0:04/0:08/0:56	2071.5
alun2	378	205	28.0	100	0:02/0:04/0:50	603.4	0:02/0:04/0:57	2750.3
rom2	504	112	25.9	66	0:04/0:06/0:58	593.7	0:04/0:10/0:45	903.5
PRNG	586	332	9.4	22	0:01/0:03/1:09	859.8	0:02/0:03/0:24	938.5
2uchp	9796	3320	131.4	0	??	O.M.	0:21/1:36/13:48	14191.2

Input format: Oct format
Hardware: Vax 3500
Time (user) reported in units of minutes:seconds, see text for format
O.M. = Out of Memory (the process aborted)

Parasitics extracted: Capacitance only

Memory reported in units of kbytes

The performance results for flat and hierarchical extractions are reported in the form: input/pre-process/final. All reported times are the user times taken by the process after the given step has completed. The input time refers to the time taken to read the layout in. Some preprocessing is done before the actual extraction; the time after the preprocessing step is reported. In flat extraction, the preprocessing step is the flattening of the layout, and in hierarchical extraction, the overlap analysis is the preprocessing step. The final time is the total time taken by the process. In other words, all times are reported relative to the start time of zero, therefore, "0:04/0:12/2:13" means the process took 4 seconds of input time, then 8 seconds for preprocessing, followed by 2:01 minutes to do the extraction.

Since the memory allocation in iCHARM was handled manually, it was possible to monitor the amount of memory that was allocated for each run of the program. The maximum amount of memory that was allocated during the process is reported in Table 3.1.

In looking over the test results in Table 3.1, some interpretations and conclusions can be formed from the data.

First of all, the overhead in the preprocessing step to handle overlap is minimal. In most cases the time to process the overlap is less than the time to flatten the entire cell. A good example of this is the *2ufrm* cell, where it took 12 seconds to flatten but only 9 seconds to process the overlap. Most cells in Table 3.1 take almost negligible time to process the overlap: from "zero" (less than 1 second) to 2 seconds.

To compare the times for hierarchical extraction versus flat extraction, it makes sense to discuss only the cells that have runtimes greater than 10 seconds. Measurement errors for the runtimes less than 10 seconds are a large percentage of the total runtime, so the shorter results cannot be trusted for comparisons. Of the results in Table 3.1, there are 18 cells where the total flat extraction time took longer than 10 seconds. These cells can be classified into three groups.

The best results show the advantages of hierarchical analysis: the hierarchical extraction takes less time and less memory. This occurs for the largest number of cells, 11 cells in all: {*mem_blk*, *ram_rowd*, *col_blk_r*, *col_blk_l*, *alur3*, *alur2*, *deccol*, *PRNGb*, *PRNGa*, *raml*, *2uchp*}. The extraction has a few overlap-rects, which means there is little or no overlap for the cell. In fact, the *mem_blk* cell has virtually no overlap; therefore, the speed-up is almost 16-fold, and one-third of the memory is used. For the *raml* and *2uchp* cells, hierarchical extraction produces an extracted result where the flat analysis failed to produce any result.

When overlap is present, the results are less impressive; five cells still have faster hierarchical extraction times but more memory is actually used: {*rom_ar2*, *data*, *ram_cold*, *rom2*, *PRNG*}. All of the cells end up having significant lists of overlap-rects. More memory must be used to calculate and create the overlap-rects. The extraction process itself is slowed since during extraction each rect for the cell must be checked to see if it touches an overlap-rect. Nevertheless, the savings due to hierarchy are great enough so that the runtime is still faster.

Finally there are two cells for which hierarchical analysis is worse in terms of both speed and memory usage: {*2ufrm*, *alun2*}. As mentioned before the padframe is the "hard" test-case for hierarchical extractors, and the *alun2* cell contains a significant amount of overlap. However, it does appear that for the 18 test-cases, most of the cells benefit from being extracted hierarchically with the methods employed by iCHARM.

4. Estimation of Current Density in the Bus

4.1 Introduction

One of the primary aims of this work is the estimation of the current density in any section of the bus (power or ground) for electromigration estimation and for re-sizing the bus for electromigration prevention, or at least increasing the MTF due to electromigration.

During the first year of this work, we developed a computer program for extracting the RC model of the bus and for computing the average current densities in the bus. The bus extractor is based on decomposing the bus geometry into geometrical primitives, such as straight, L, and T primitives. Each primitive is pre-characterized using finite-element-methods. During extraction of the bus the RC model of each extracted primitive is used. A linear solver is then used to solve for the average current density in any section of the RC bus nodal, given the average current density at each contact node as obtained from the probabilistic simulator.

During the second phase of this work, we improved the analysis of the bus in three directions. First, we completed the modeling of the contacts. Strips of metal containing contacts were analyzed using the finite element method for various combinations of contact sizes and metal widths, and the result store in the set of primitives.

Second, we have shown that MTF due to electromigration depends not only on the average current density, but also on the variance of the current density waveform [2]. For example, two density waveforms having the same average, but different variance may lead to different MTF estimation. This fact has been observed experimentally and reported in literature.

The third issue considered is the computation of the variance current density waveform in the bus, in addition to the average current density waveform. Recall that the variance current waveforms drawn by the logic gates at the bus contacts are computed by the probabilistic simulation, in addition to the average current waveform. Earlier, we have only concentrated on the computation of the average current density [5] and the calculation of the variance was not done. In this work we have developed algorithms and software for efficient computation of both the average and the variance current densities in the bus.

The problem of finding average and variance of branch currents in a general RC network was first published by Najm et. al. in [2]. The method involves finding the infinite impulse response functions for each of the branch currents of the network due to unit current impulses applied at contact points, one at a time. To calculate the variance of branch currents, one needs to carry out a convolution operation involving infinite terms. Such an analysis is extremely complex and perhaps not necessary when the capacitive effects of buses are almost negligible [9]. Even for simple resistive networks, the above method involves inverting the node admittance matrix, which is very expensive both from computation and storage points of view. In this paper, we have simplified the above analysis for resistive networks and have made it very efficient. The sparsity of the node admittance matrix is exploited to reduce CPU time and memory space requirements.

We assume that there are *n* nodes and *b* branches in the resistive network that represents the bus. Let Ib denote the vector of branch currents, Is the vector of currents entering the nodes (contact point currents) and Vn the vector of node voltages. We denote j^{th} element of a vector V by V_j and the i,j^{th} element of a matrix A by A_{ij}. The i^{th} column vector of a matrix A will be denoted by $[A]_i$. The transpose of a matrix A will be denoted by A^T. We assume that the contact point current waveforms (both average E[Is] and variance Var[Is]) have already been sampled in time and there are *m* time points for which the network needs to be analyzed. Furthen ore, as justified in [1], we assume that various contact point currents are mutually independent at a given time point.

4.2 Networks containing no loops (TREE):

For a network with no loops, current in a branch (say kth one) can be expressed as a linear combination of the leaf (contact) currents:

$$Ib_k = \sum_{j=1}^{n} h_{jk}\, Is_j \tag{4.1}$$

where $h_{jk} \in 0,1$. $h_{jk} = 1$ if node *j* is a descendent of branch *k* and $=$ 0 otherwise. By taking expectation, we get the average value of the branch current:

$$E[Ib_k] = \sum_{j=1}^{n} h_{jk}\, E[Is_j]. \tag{4.2}$$

Since various contact currents are assumed to be mutually independent, we can write:

$$Var[Ib_k] = \sum_{j=1}^{n} h_{jk}^2\, Var[Is_j] = \sum_{j=1}^{n} h_{jk}\, Var[Is_j]. \tag{4.3}$$

Therefore, by analyzing the tree is in a bottom up fashion (from contact points up to the pad), we obtain the average and the variance of currents in various branches. On a serial computer, this algorithm would take O(mn) time in the worst case.

4.3 Networks containing loops (GENERAL):

For a general resistive network, to calculate branch currents, we need to calculate node voltages first. The following equation holds between Vn and Is, where Yn is the *node admittance matrix* of the network:

$$Yn\ Vn = Is. \tag{4.4}$$

By taking expectation:

$$Yn\ E[Vn] = E[Is]. \tag{4.5}$$

From the LU factors of the Yn matrix, this equation can be solved to obtain the average values of node voltages. If we assume that ith resistor (conductance = y_i) is connected between nodes l and k in the network, then

$$E[Ib_i] = (E[Vn_l] - E[Vn_k]) \times y_i. \tag{4.6}$$

Thus, we can calculate the average values of currents in all the branches. The calculation of variance is more complex. We define the following covariance matrices:

$$KIb \triangleq E[(Ib - E[Ib])(Ib - E[Ib])^T \tag{4.7a}$$

$$KIs \triangleq E[(Is - E[Is])(Is - E[Is])^T] \tag{4.7b}$$

$$KVn \triangleq E[(Vn - E[Vn])(Vn - E[Vn])^T] \tag{4.7c}$$

The diagonal entries of KIb are the required variance entries of interest. As before, if the ith resistor (conductance = y_i) is connected between nodes l and k, then variance of current in the resistor is:

$$KIb_{ii} = (KVn_{ll} + KVn_{kk} - 2KVn_{lk}) \times y_i^2. \tag{4.8}$$

Thus, to calculate the diagonal entries of KIb, only (n+b) elements of KVn matrix are needed. Elements needed are the *n* diagonal elements and *b* off diagonal elements (KVn_{lk}) corresponding to each resistor between nodes l and k. (Note $KVn_{lk} = 0$ if l or k is the reference node.)

To find the required KVn entries, we proceed as follows. Eq. (4.4) can be written as Vn = Rn Is. Rn is the *resistance matrix*, which is inverse of Yn. By substituting this in Eq. (4.7c), we get the following relationship between KVn and KIs:

$$KVn = Rn\ KIs\ Rn. \tag{4.9}$$

Note that Rn is a symmetric matrix. KIs is a diagonal matrix because of the assumption that various contact currents are mutually independent. All of the diagonal entries of KIs are positive, ith entry being $Var(Is_i)$ (= σIs_i^2 say). Thus, KIs is a positive definite matrix and therefore it has a positive definite diagonal square root (call it σIs):

$$KIs = \sigma Is\ \sigma Is = (\sigma Is)^2. \tag{4.10}$$

The ith diagonal entry of σIs is σIs_i. This equation can be used to express KVn matrix as follows (since Rn is symmetric):

$$KVn = (Rn\ \sigma Is)(Rn\ \sigma Is)^{T.} \tag{4.11}$$

Let $\sigma Vn = Rn\ \sigma Is$. Note that σVn is not symmetric. Equation (4.11) becomes:

$$KVn = (\sigma Vn)(\sigma Vn)^{T.} \tag{4.12}$$

If we know σVn, we can calculate scriptl,kth entry of KVn by the equation:

$$KVn_{lk} = \sum_{j=1}^{n} \sigma Vn_{lj} \times \sigma Vn_{kj}. \tag{4.13}$$

Since Yn is the inverse matrix of Rn, we can rewrite $\sigma Vn = Rn\ \sigma Is$ as:

$$Yn\ \sigma Vn = \sigma Is. \tag{4.14}$$

For each of the *n* corresponding columns of σVn and σIs matrices, we get *n* matrix equations of the form: $(i = 1, 2, \cdots n)$

$$Yn \begin{bmatrix} \sigma Vn_{1i} \\ \sigma Vn_{2i} \\ \vdots \\ \sigma Vn_{2i} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ \sigma Is_i \\ \vdots \\ 0 \end{bmatrix} \quad \text{or} \quad Yn[\sigma Vn]_i = [\sigma Is]_i. \tag{4.15}$$

One can solve these n equations and obtain the required KVn entries from Eq. (4.13). However, the node admittance matrix of a resistive network is usually very sparse and therefore, even for large values of n, it can be stored very compactly on a computer. On the other hand, σVn matrix is usually a full matrix. Therefore, for large values of n, storage would be a problem. However, from Eq. (4.13), we notice that to calculate KVn_{lk}, at any time, we do not need to multiply two entries of σVn matrix from two different columns. Therefore, we do not need to compute and store the entire σVn matrix at any time. As we calculate columns of σVn, we perform the multiplications needed in Eq. (4.13) for all the required KVn_{lk} entries and then discard the column vector. If we assume that the solution phase (backward and forward substitutions) requires $O(n^{\alpha})$ multiplications then to calculate average and variance current waveforms, we need $O(n^{1+\alpha} + mn^2 + mn(n+b))$ multiplications. Computational speedup can be obtained by using parallel processing since the above algorithm is quite suitable for implementation on a medium to coarse grain parallel machine.

4.3 Experimental Results

In this section, we demonstrate the effectiveness of the various algorithms described in the previous section on several examples. All of the experiments reported in this section were carried out on a VAX 3500 computer. Since the main emphasis of this part of the work has been to come up with efficient algorithms for the calculation of average and variance of branch currents, we have randomly generated several resistive networks and have assigned random values for average and variance of contact currents. In Table I, we report results of the experiments for TREE networks. The pre-processing phase consists of generation of the network and setting up of various data structures. Table II shows similar results for GENERAL networks. For the LU decomposition and the solution phase, we have used the sparse linear equation solver package [10].

The details of the derivation of the above solution technique, with examples, are given in Ref. [6].

Table I: Processing of TREE Networks

n	m	CPU time in (sec) needed for		
		pre-processing	average	variance
100	10	0.03	0.02	0.02
	50	0.25	0.05	0.05
500	10	0.32	0.07	0.05
	50	1.17	0.27	0.28
1000	10	0.62	0.13	0.12
	50	2.42	0.57	0.55
2000	10	1.32	0.25	0.25
	50	4.60	1.15	1.12
5000	10	3.02	0.62	0.63
	50	11.68	2.83	2.82

Table II: Processing of GENERAL Networks

n	b	m	CPU time (in sec) needed for		
			pre-processing	averge	variance
10	12	10	0.02	0.02	0.05
		50	0.03	0.05	0.22
	20	10	0.02	0.03	0.05
		50	0.05	0.07	0.30
50	60	10	0.13	0.05	1.08
		50	0.20	0.30	5.07
	100	10	0.18	0.10	1.52
		50	0.32	0.42	7.08
100	120	10	0.27	0.12	4.30
		50	0.40	0.63	20.32
	200	10	0.57	0.20	6.27
		50	0.77	1.00	28.90
500	600	10	1.97	0.72	113.40
		50	2.62	3.58	521.72
	1000	10	15.18	2.02	205.28
		50	14.85	9.65	776.35

V. Summary

In this report, we have summarized our accomplishments over the past year on the problem of computing current density statistics in CMOS circuits for reliability estimation. The emphasis has been on the use of the average value and variance of the current density to estimate MTF due to electromigration in the bus. The work involves three subtasks: (1) extraction from layout; (2) probabilistic simulation of transistor circuit to compute the average value and variance of current drawn by the circuit at contact points; and (3) the computation of the current density in the bus and estimation of the MTF. An overview of the system that we developed is given in [7].

In the work on extraction, we have developed a hierarchical extractor based on our previous extractor. The new extractor preserves the hierarchy inherent in the design, thus allowing hierarchical simulation and saving storage. In probabilistic simulation, we have extended and enhanced the accuracy of our initial derivation and started the development of a hierarchical probabilistic simulator that exploits the hierarchy in the design and the data structure. For the computation of the current density in the bus, we have developed circuit reduction and sparse matrix techniques to compute the average value as well as the variance current density. The RC circuit model of the bus, although linear, is very large that could contain more than one hundred thousand nodes. Special numerical techniques are then necessary to handle the complexity of the computation.

So far, we have concentrated our efforts on estimating the current density for estimating MTF due to electromigration. However, we believe that the probabilistic approach is applicable to other reliability studies as well. We have thus initiated two projects in that direction. One is the estimation of hot-carrier effects in transistors. Hot-carrier degradation is also a long-term reliability issue that is cumulative in nature, depending on how long a transistor operates in saturation under all possible primary input conditions, and thus depends on its location in the circuit. Exhaustive simulation would be exorbitantly expensive to estimate hot-carrier effects. We are currently developing probabilistic techniques to estimate these effects with reasonable computational effort. We are also developing techniques to estimate voltage drop in the busses.

Voltage drop is also becoming of concern with the decrease in feature size. It will become more critical when the supply voltage is reduced and when BICMOS circuits are used in the design since BICMOS circuits draw a larger current during switching that purely CMOS circuits. In contrast to electromigration and hot-carrier effects, voltage drop is a worst-case phenomenon, and our aim is to develop techniques to sort out these worst-case situations. Our ultimate aim is to optimize a given design, or give design guidelines, to keep electromigration and voltage drop within acceptable bounds everywhere in the bus, as well as keep degradation with transistors caused by hot-carrier effects within operating limits, for all possible inputs.

Publications and References

[1] F. Najm, R. Burch, P. Yang, and I. N. Hajj, "Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits," *IEEE Transactions on Computer-Aided Design*, Vol. 9, No. 4, pp. 439-450, April 1990.

[2] F. N. Najm, I. N. Hajj, and P. Yang, "An Extension of Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits," accepted for publication in the *IEEE Trans. on Computer-Aided Design*, October 1991.

[3] G. Stamoulis, "New Techniques for Probabilistic Simulation of VLSI CMOS Circuits," M.S. Thesis, Department of Electrical and Computer Engineering, University of Illinois, September 1991.

[4] Russell M. Iimura, "iCHARM: Hierarchical Circuit Extraction Including Interconnect Parasitics," M.S. Thesis, Department of Electrical and Computer Engineering, University of Illinois, August 1990.

[5] Hungse Cha, "Current Density Calculation Using Rectilinear Region Splitting Algorithm for Very Large Scale Integration Metal Migration Analysis," M.S. Thesis, Department of Electrical and Computer Engineering, University of Illinois, August 1990.

[6] H. Kriplani and I. N. Hajj, "Calculation of Currents in P/G Busses for Electromigration Analysis," University of Illinois, Coordinated Science Laboratory, Technical Report, May 1991. Also, to be published in the *Proceedings of the Internationl Symposium on Circuits and Systems*, San Diego, CA, May 1992.

[7] I. N. Hajj, V. B. Rao, R. Iimura, H. Cha, and R. Burch, "A System for Electromigration Analysis in VLSI Metal Patterns," *Custom Integration Circuits Conference*, San Diego, CA, May 12-15, 1991.

[8] Athanasios Papoulis. *Probability, Random Variables and Stochastic Processes*, 2nd edition: New York, McGraw-Hill Book Co. 1984.

[9] S. Chowdhury and M. A. Breuer, "Optimum Design of IC Power/Ground Nets Subject to Reliability Constraints," *IEEE Trans. on Computer-Aided Design*, Vol. 7, no. 7, pp. 787-796, July 1988.

[10] K. S. Kundert and A. Sangiovanni-Vincentelli, "Sparse User's Guide: A Sparse Linear Equation Solver," Department of Electrical Engineering and Computer Science, University of California, Berkeley, version 1.3a, April 1988.